A Computationally Efficient Scheme for Hierarchical Predictive Control

Kaustubh Pathak, Ph.D Student *
Department of Mechanical Engineering
University of Delaware, Newark DE 19716.

Sunil Kumar Agrawal, Ph.D, Professor †
Department of Mechanical Engineering
University of Delaware, Newark DE 19716.

Elena Messina,
National Institute of Standards and Technology,
Gaithersburg, MD 20899

Abstract

A real-time hierarchical scheme for predictive control of nonlinear systems is formulated. Different levels in the hierarchy operate asynchronously while using data buffers to intercommunicate. They run at different sampling rates and have different planning horizons but use the same underlying algorithm based on finite element collocation and *Sinc* function interpolation. After an initial planning phase, the system starts running, and the system state feedback is utilized by each planning layer to refine the plan of its predecessor but for a shorter planning horizon. The results of this approach is applied to a nonlinear simplified kinematic model of a mobile robot. The algorithm is implemented using the NIST RCS Framework for real-time control.

Keywords: *Model predictive control, Hierarchical Control, Sinc Interpolation, Numerical Dynamic Optimization.*

1 Introduction

Model Predictive Control (MPC) is a practical online control methodology where the system being controlled operates under constraints and an optimal control strategy is not available in an analytical form. MPC has been successfully applied in chemical industry where the system dynamics is typically much slower than that found in mechanical systems like mobile robots.

MPC involves solving a finite horizon optimization problem online at each sampling time instant, using the current system states as the initial conditions. The control action so generated is not globally optimum, nor does it guarantee stability a priori. Some recent papers (refer [2] for a survey) have focused on proving stability using Dynamic Programming and Control Lyapunov Function approaches.

For application of nonlinear MPC methods to mechanical systems with faster dynamics, the computation time and effort involved in solving the optimization problem at each time instant becomes significant. The problem of finite computation time has not been studied extensively for nonlinear MPC [4]. Typically, the sampling intervals need to be tuned to obtain a compromise between performance and the computation power available. Smaller sampling times offer more accurate predictions but with the same planning horizon, the dimension of the optimization problem increases. Larger sampling intervals, however, fail to ensure satisfaction of constraints during the prediction horizon and may not capture fast system dynamics and thus lose performance.

This paper formulates the MPC problem into a hierarchy of predictive controllers each with its own prediction horizon and sampling rate. This arrangement is shown to alleviate some of the aforementioned problems arising out of the finite computational time required to solve the problem. A simplified nonlinear kinematic model of a unicycle like mobile robot is used to illustrate the strategy.

The paper is organized as follows: section 2 mathematically formulates the basic MPC problem and discusses the issue of how to obtain an efficient discretization of the problem at hand. The choice of a suitable interpolation function set is discussed in subsection 2.2, and the suitability of *Sinc* interpolation for predictive planning is pointed out. Section 3 discusses the idea behind the proposed scheme of hierarchical controllers and the basic algorithm applied at each planning layer is given in subsection 3.1. The software tools used to implement the algorithm are mentioned in section 4. Section 6 describes the application of this method to

*pathak@me.udel.edu
†Corresponding author: agrawal@me.udel.edu

a real-time simulation of a unicycle like mobile robot using nonlinear kinematic equations. Finally, conclusions and directions of future work are presented in the conclusions.

2 Problem Formulation

Let the system be represented by a nonlinear time-invariant equation:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \;\; \mathbf{x} \in \Re^n, \; \mathbf{u} \in \Re^m \tag{1}$$

where $\mathbf{x}$ is the state vector and $\mathbf{u}$ is the input vector. Let t_i be the current sampling time and let $\mathbf{x}(t_i)$ be the current system state. Let ΔT be the sampling period and let there be N such periods in the planning horizon. Then the end-point of the planning horizon is at $t_N = t_i + N\Delta T$. The nonlinear programming (NLP) problem to be solved for the horizon $t \in [t_i, t_N]$, for the predictive controller is:

$$\underset{\mathbf{u}(t)}{Min} J_N(\mathbf{u}(t), \mathbf{x}(t_i)) = \int_{t_i}^{t_N} L(\mathbf{x}, \mathbf{u})dt + F(\mathbf{x}(t_N)) \tag{2}$$

subject to:

$$\mathbf{u}(t) \in \Upsilon, \; \mathbf{x}(t_i) = \mathbf{x_0}, \; \mathbf{x}(t_i + N\Delta T) \in \mathbf{X_f}$$

where J_N is the objective function for the planning horizon, and is comprised of a terminal cost F and an integral cost L. *Note that this problem is solved for each sampling instant t_i.* Some authors distinguish between a planning horizon $P\Delta T$ and a control horizon $M\Delta T$, $(M \leq P)$, and the planned control is only applied during the control horizon.

In this paper, the control horizon is defined in terms of another parameter ΔT_c, which is the average computational time required to solve the NLP problem. For the system to always have a plan while it is running, the condition $P\Delta T \geq \Delta T_c$ should hold. We consider the control horizon $M\Delta T$ to be the same as ΔT_c, i.e. the input plan is updated as soon as it is computed. To satisfy the above condition, an initial tuning phase is normally needed. We further restrict our attention to a two point boundary value problem where, $t_0, \mathbf{x}(t_0), t_f, \mathbf{x}(t_f)$ are given. This problem is relevant to mobile robots where a set of obstacle-free way-points might be specified by a higher level planner along with a total time to traverse the path.

A commonly used method for solving the above NLP is to use the method of orthogonal collocation on finite elements [11], where a certain set of system variables is discretized over the planning horizon. Let $\mathbf{y}(t), \mathbf{y} \in \Re^p$ be a vector variable being discretized, then one defines the following terms:

$$\mathbf{y}_{k|i} = \mathbf{y}(t_i + k\Delta T), \; k \in [0, N] \tag{3}$$

$$\mathbf{Y}(i, \Delta T, N) \equiv \left[\mathbf{y}_{0|i} \ldots \mathbf{y}_{N|i}\right]_{p \times (N+1)}$$
$$\mathcal{T}(t_i, \Delta T, N) \equiv \{t : t \in [t_i, t_i + N\Delta T]\}$$

If the variable $\mathbf{y}(t)$ represents the state or the input, one can define a $n \times (N+1)$ matrix $\mathbf{X}(i, \Delta T, N)$ and a $m \times (N+1)$ matrix $\mathbf{U}(i, \Delta T, N)$ respectively. One also defines:

$$\mathbf{I}(t) = [I_0(t) \ldots I_N(t)]^T, \; t \in \mathcal{T}(t_i, \Delta T, N) \tag{4}$$

as the interpolating function used within the planning horizon. For a given variable $\mathbf{y}(t), \mathbf{y} \in \Re^p$ to be discretized and interpolated within the planning horizon, one can write:

$$\mathbf{y}(t) = \mathbf{Y}(i, \Delta T, N)\, \mathbf{I}(\mathbf{t}), \;\; t \in \mathcal{T}(t_i, \Delta T, N) \tag{5}$$

2.1 What to Discretize?

2.1.1 General Systems

For a general system given by Eq.(1), one needs to discretize all states and inputs. The NLP Eq.(2) is then reduced to:

$$\underset{\mathbf{U}(i, \Delta T, N)}{Min} J_N(\mathbf{U}(i, \Delta T, N), \mathbf{x}_{0|i}) =$$
$$\mathbf{L}[\mathbf{X}(i, \Delta T, N), \mathbf{U}(i, \Delta T, N)] \int_{t_i}^{t_N} \mathbf{I}(t)dt + F(\mathbf{x}_{N|i}) \tag{6}$$

subject to:

$$\mathbf{u}_{j|i} \in \Upsilon, \; j \in [0, N], \; \mathbf{x}_{0|i} = \mathbf{x}(t_i), \; \mathbf{x}_{N|i} \in \mathbf{X_f} \tag{7}$$

$$\mathbf{X}(i, \Delta T, N)\frac{d\mathbf{I}(t)}{dt} = \mathbf{f}(\mathbf{X}(i, \Delta T, N), \mathbf{U}(i, \Delta T, N)) \tag{8}$$
$$t \in \{t_c\} \subset \mathcal{T}(t_i, \Delta T, N)$$

where the dynamic equations are applied as nonlinear equality constraints (Eq.8) at time instants given in the set $\{t_c\}$, to the problem. The dimesion of the problem is now $(N+1) \times n + (N+1) \times m$. Note also that in Eq.(8) one has to differentiate the interpolating function, which in general gives poor approximation near the boundary points.

2.1.2 Differentially Flat Systems

If the nonlinear system has the property of differential flatness(this includes feedback linearizable systems [9] [10]), one can reduce the dimension and complexity of the NLP considerably. The system 1 is differentially flat if one can find a set of outputs $\mathbf{z}$, $\mathbf{z} \in \Re^m$, such that all states and inputs can be represented in terms of $\mathbf{z}$ and its derivatives:

$$\mathbf{x} = \mathbf{x}(\mathbf{z} \ldots \mathbf{z}^{(p)}), \; \mathbf{u} = \mathbf{u}(\mathbf{z} \ldots \mathbf{z}^{(p)}) \tag{9}$$

Therefore instead of discretizing states and inputs, one can just discretize the highest derivative of the flat output $\mathbf{z}^{(p)}$ in the planning horizon. Then the following recursive strategy could be used to get all other derivatives of $\mathbf{z}(t)$

$$\mathbf{z}^{(l)} = \mathbf{Z}_l(t_i, \Delta T, N)\, \mathbf{I}(t), \quad l \in [1, p] \tag{10}$$

$$\mathbf{z}^{(l-1)}(t) = \mathbf{z}^{(l-1)}(t_i) + \mathbf{Z}_l(t_i, \Delta T, N) \int_{t_i}^{t} \mathbf{I}(t)dt \tag{11}$$

$$t \in \mathcal{T}(t_i, \Delta T, N)$$

$$\mathbf{Z}_{l-1}(t_i, \Delta T, N) \equiv [\mathbf{z}_{j|i}^{(l-1)}]_{m\times(N+1)},\ j \in [0, N] \tag{12}$$

where Eq.(12) is obtained by sampling Eq.(11). The initial conditions $\mathbf{z}^{(j)}(t_i),\ j \in [0, p-1]$ can be obtained from the corresponding state initial conditions by substitution in Eq.(9). One then obtains all discretized states and inputs in terms of the discretized values of $\mathbf{z}^{(p)}$, i.e. $\mathbf{X}[\mathbf{Z}_p(t_i, \Delta T, N)],\ \mathbf{U}[\mathbf{Z}_p(t_i, \Delta T, N)]$. The above can now be substituted in the NLP problem (Eq.6) which now has the reduced order of $(N+1)\times m$. Note that one has the added advantage of only integrating the interpolating function (which gives good results in practice) and taking care of the initial conditions explicitly. Also, the dynamic equations are implicitly satisfied and therefore need not be applied as nonlinear equality constraints to the problem.

2.2 Choice of Interpolation $\mathbf{I}(t)$

The most common choice is to use various kinds of polynomials [11]. Lagrange polynomials with uneven collocation points were used by ([5]) where the collocation points are selected as the roots of shifted Legendre polynomials for accurate integration, i.e.

$$I_i(t) = L_i^N(t) = \prod_{j=0,\ j\neq i}^{N} \frac{t - t_j}{t_i - t_j} \tag{13}$$

A problem with polynomial interpolation is that as the number of points increases, the interpolation becomes oscillatory between the collocation points. This causes convergence problems during the solution of the NLP. It also introduces high frequency ($\geq$ Nyquist frequency $1/2\Delta T$) components in the 'signals' $\mathbf{x}(t)$ and $\mathbf{u}(t)$ whose true behaviour cannot be discovered by 'sampling'(collocating) at an interval of ΔT. The ideal interpolation formula from signal processing [7] states that an analog signal $\mathbf{x}(t)$ can be recovered from its samples $\mathbf{x}(i\Delta T)$ at sampling periods ΔT, if the signals bandwidth B is such that: $1/\Delta T \geq 2B$. If the above condition is satisfied then the ideal interpolation function is given by a particular form of Sinc(Cardinal) function:

$$I_i(t) = S_i(t, \Delta T) = \frac{\sin(\pi(t - i\Delta T)/\Delta T)}{\pi(t - i\Delta T)/\Delta T} \tag{14}$$

where the *Sinc* function $sinc(x) \equiv \sin(\pi x)/(\pi x)$ is defined to be 1, if $x = 0$. The following properties of the Sinc interpolation are briefly noted ([6], [8]):

1. $\lim_{N\to\infty} L_i^N(t) = sinc(t - t_i)$, where L_i^N is the Lagrange polynomial introduced in Eq.(13) and t_i is the i-th sample instant [8]. Similar results can be obtained for Spline interpolations.

2. For a particular sampling period (collocation interval) ΔT, the Sinc interpolation $S_i(t, \Delta T)$ would make sure that frequencies higher than $1/(2\Delta T)$ are not present in the interpolated variable.

3. In the methodology introduced in section 2.1.2, we need to compute the integral of $I_i(t)$. If $I_i(t) = S_i(t, \Delta T)$, there exist efficient algorithms for evaluating this integral [12].

3 Computational Algorithm

Using $I_i(t) = S_i(t, \Delta T)$, as introduced above, one can apply the idea of hierarchical multi-resolution planning [1], to setup a hierarchy of predictive controllers. Let ΔT be the fundamental sampling period of the system. Let $D_p(\Delta T)$ denote a set of functions representable by $S_i(t, p\Delta T)$, i.e, they are bandlimited by $1/(2p\Delta T)$ Hz. Clearly, if $q \leq p$, then $D_p(\Delta T) \subset D_q(\Delta T)$. Let the initial desired trajectory be given from $\mathbf{x}(t_0 = 0)$ to $\mathbf{x}(t_f)$. This is assumed to have been planned by the highest level planner at level 0. This trajectory may or may not satisfy the system's dynamic equations and other constraints and could be in the form of a suitably interpolated way points, which traverse an obstacle free path as in case of a mobile robot. Planner 0 computes this trajectory offline before the system starts running. This solution is written to a data- buffer B_1. Then the following algorithm is used for planning levels $l = 1 \ldots (N_h - 1)$, where the last level N_h is the system being controlled.

3.1 Algorithm for Planner at level l

In this section, $n_1 > n_2 > \ldots > n_{N_h-1}$ are chosen integers. Each predictive level l ($l \in [1, N_h - 1]$)does its planning in H intervals, and its sampling period (collocation interval) is $n_l\Delta T$ seconds, where ΔT is the fundamental sampling period of the physical system. Level l does the following *at each executution instant* t_i:

1. If $t_i + n_l H\Delta T > t_f$ stop. The time-to-go is less than the planning horizon of this level, so it stops.

2. Take the plan in Buffer B_l. Buffer B_l contains a set of sampled states $\mathbf{X}(j, n_{l-1}\Delta T, H)$ along with

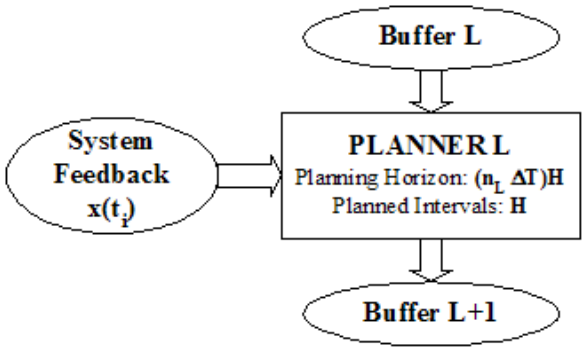

Figure 1: The Predictive Planner at Level L

the sampling period $n_{l-1}\Delta T$ and the time t_j the plan was generated, of the last plan generated by Planner level $l-1$. Interpolate these stored values using $\mathbf{S}(t, n_{l-1}\Delta T)$, to obtain the value of the state $\mathbf{x}$ at $t = t_i + n_l H \Delta T$, i.e till the end of this level's horizon. This becomes the goal point of the new plan that this level is about to generate. Read the current system state $\mathbf{x}(t_i)$ from the feedback buffer.

3. Solve the NLP as described in section 2.1.2. Use the interpolation function vector $\mathbf{S}(t, n_l\Delta T)$, for the planning horizon $\mathcal{T}(t_i, n_l\Delta T, H)$ using $\mathbf{Z_p}(i, n_l\Delta T, H)$ as parameters. The solution provides an input which is 'optimum' for the above horizon with respect to the domain $D_{n_l}(\Delta T)$, as defined above. For this period, this solution is better than the solution generated by the previous level, as $D_{n_{(l-1)}}(\Delta T) \subset D_{n_l}(\Delta T)$. Each level thus, shortens the planning horizon but increases the frequency resolution.

4. Let this solution be $\mathbf{X}(j, n_l\Delta T, H)$. Interpolate the rest of the solution from B_l in the time-range $[t_i + n_l H\Delta T, t_f]$ using $\mathbf{S}(t, n_l\Delta T)$, and append it to the solution generated from the current planning. Write this concatenated full planned solution ($[t_i, t_f]$) in solution Buffer B_{l+1}.

When the system at level N_h receives its first plan from level $N_h - 1$, it synchronizes its time with the planned time and starts running. The above identical algorithm is run by each level at each instant of execution time. An example structure for the case $N_h = 3$ is shown in figure 2.

The following points are to be noted:

1. At any instant, the system has a full plan from t_0 to t_f, however the solution near the current time t_i has a much finer resolution and therefore a better quality. The danger of the computation time exceeding the planning horizon is somewhat mitigated by this.

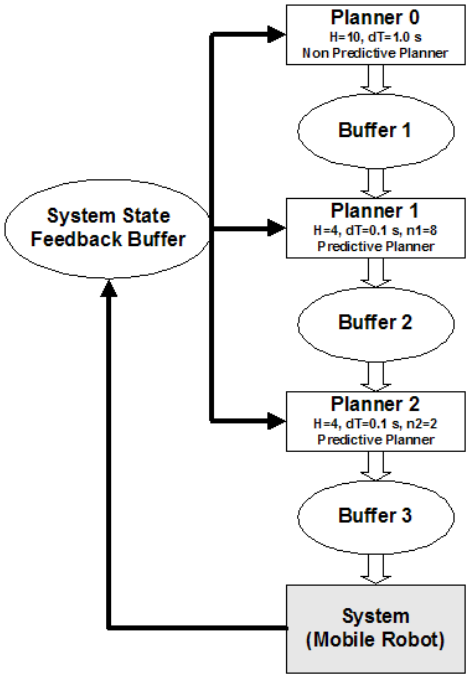

Figure 2: The hierarchy of the predictive planners

2. The planned state vector $\mathbf{x}(\mathbf{t})$ is smooth within a planner's horizon. At the boundary of the solutions of two levels the $\mathbf{x}(\mathbf{t})$ is C^0 continuous. The planned input $\mathbf{u}(\mathbf{t})$ is in general discontinuous at the level boundaries, as increasing the frequency resolution might give an optimum which is very different from the last generated solution. Therefore each solution buffer contains only the planned state samples not the inputs. The planned inputs can be obtained from back-substitution of the state-interpolation in the dynamic equation. This effect can be reduced if the solution of level $l-1$ is given as an initial guess to level l. Then the liklihood of a local minimum being found around the previous solution increases.

3. All levels are running asychronously with respect to each other. They communicate only via shared data buffers.

4 Implementation

The above algorithm fits in naturally with the Real-Time Control System (RCS) Library [13] developed by NIST. Different levels are designed as seperate processes running on a Linux machine and communicating through shared-memory data buffers. The NLP at each level is solved by the SQP algorithm as implemented in NPSOL. A C++ wrapper for NPSOL library is provided by Sandia National Laboratories [14] which

is used for the purpose.

5 Simulation

The idea presented in this paper is illustrated by a simulation of a simplified nonlinear kinematic model of a unicycle like mobile robot. The kinematic equtions for the system are:

$$\dot{x} = v\cos(\theta),\ \dot{y} = v\sin(\theta),\ \dot{\theta} = \omega \tag{15}$$

The inputs to this system are $\mathbf{u} = [v(t), \omega(t)]$, where $v(t)$ the forward heading speed in meters/sec, and $\omega(t)$ is the rotational speed in rad/sec. The states are $\mathbf{x(t)} = [x, y, \theta]$, where x and y are planar coordinates in an inertial frame and θ is the orientation of the robot with respect to the x axis. The constraints on the system are:

$$-2.5 \leq v(t) \leq +2.5,\ \ -4.71 \leq \omega(t) \leq +4.71 \tag{16}$$

Since the time behaviour is already specified in the problem, the cost is specified as the minimum input problem for each level:

$$\underset{\mathbf{u}(t)}{Min}\ J_N(\mathbf{u}(t), \mathbf{x}(t_i)) = \int_{t_i}^{t_i+N\Delta T} (k_1 v^2 + k_2 \omega^2) dt \tag{17}$$

k_1 and k_2 are scaling factors and are selected as 1.0. The final state $\mathbf{x}(t_i + N\Delta T)$ for each level is generated from the solution of the previous level as mentioned in section 3.1. The set $\mathbf{X_f}$ for each level is a hypercube around this final point which is specified in terms of the final state error allowable. This allowable error is bigger in the θ dimension ($\pm\pi/4$) than in the x and y dimension (0.01 m). The flat ouputs of this system are $[x, y]$, but we chose to directly discretize the inputs v, ω, as the cost and constraints and specified direcly in terms of them. The discretized version of the above problem can be stated as follows:

$$\underset{\mathbf{V}, \mathbf{\Omega}}{Min}\ J_N(\mathbf{x}_{0|i}) = \left[\mathbf{V}^2 + \mathbf{\Omega}^2\right]_{1\times(N+1)} \int_{t_i}^{t_N} \mathbf{S}(t, \Delta T) dt \tag{18}$$

where $\mathbf{V}^2 = \mathbf{V}^2(i, \Delta T, N)$ and $\mathbf{\Omega}^2 = \mathbf{\Omega}^2(i, \Delta T, N)$ are the samples of the corresponding elementwise squared values. The gradient and Hessian of the above cost function with respect to the parameters $[v_0, \ldots, v_N, \omega_0, \ldots \omega_N]$ are required for a fast execution of the NLP SQP algorithm. These can be directly computed from the above expression.

Three levels of planners, and a robot simulator are designed:

1. The 0th level planner is given an initial $\mathbf{x_0} = \mathbf{0}$ and a final point $\mathbf{x_f} = [-5, 2.5, \pi]$, and a desired runtime $t_f = 10$ seconds. This planner runs initially once and creates a trajectory.
2. Level 1 and 2: These planners run according to the algorithm in section 3.1.
3. The robot simulator runs at the frequency of 10 Hz. It uses timers to synchronize simulation time with real-time. It uses a high resolution Euler type integrator to integrate its state based on the plans it receives in buffer B_3. A small disturbance term is added to this integrator to test the effectiveness of MPC.

The results are shown in figures 3- 5.

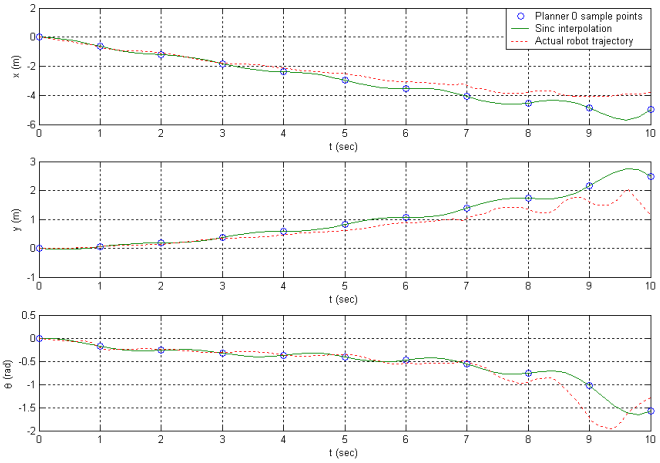

Figure 3: Simulation: The trajectory planned by the non-predictive planner 0, and the actual robot trajectory (dotted).

The discrepancy between planned and actual simulated robot teajectory can be attributed to inaccuracies in Euler integration along with a difficulty in synchronizing the simulation time with real-time exactly.

6 Experiment

An experiment based on the simulation of Sec.5 was conducted using an IRobot's Magellan Pro Robot. The setup of the planner hierarchy is the same as in Sec.5. A typical run is depicted in Fig.6. The planner 3 (the last planner) plan which acts as a set point for the robot at each instant is shown along with the trajectory of the robot. It shows that MPC is robust with respect to errors in the initial position of the robot.

7 Conclusion

This paper presents a hierarchical predictive controller which works despite finite computation times and relatively fast system dynamics. This is achieved by computing a multiresolutional (in time and frequency) overall plan for the system, which is always most accurate

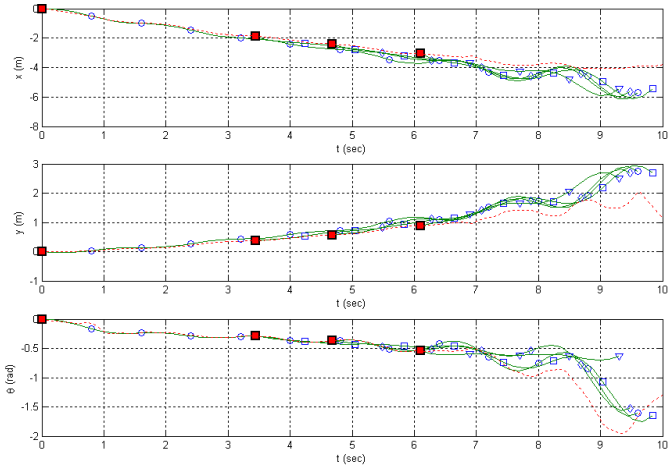

Figure 4: Simulation: The trajectory planned by the predictive planner 1 having a planning horizon of 3.2(= 4×0.8) sec at four time instants. The filled color box represents the starting sample of the robot position which was used to plan. Only the first 3.2 sec of each trajectory has been planned by planner 1. The rest is the plan of higher level planners interpolated at planner 1 sampling frequency. The dotted line is the actual robot trajectory. The trajectories do not end at $t_f = 10$, if the time-to-go is not an integer multiple of the sampling period of planner 1. Note that there is more variance in the final θ state achieved than for x and y. The dotted curve is the actual robot trajectory.

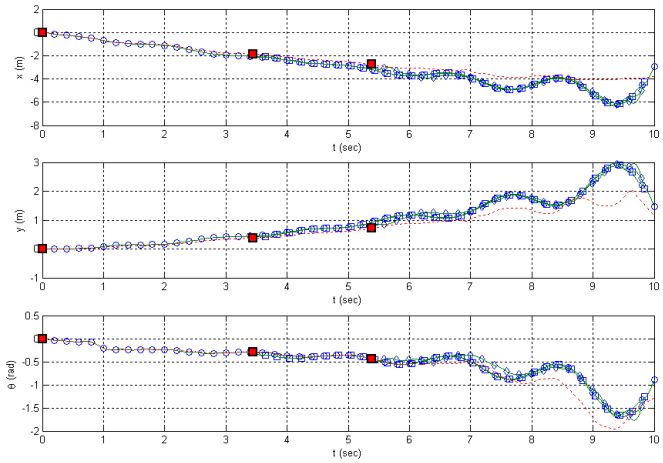

Figure 5: Simulation: The trajectory planned by the predictive planner 2 having a planning horizon of 0.8(= 4×0.2) sec at three time instants. The filled color box represents the starting sample of the robot position which was used to plan. Only the first 0.8 sec of each trajectory has been planned by planner 2. The rest is the plan of higher level planners interpolated at planner 2 sampling frequency. The dotted curve is the actual robot trajectory.

near the current time. The system is implemented using the RCS library and its efficacy is tested in a real-time simulation of a nonlinear kinematic model of a mobile robot.
Further work is underway in terms of testing the framework in higher dimensional dynamic models and application to a real mobile robot environment.

8 Acknowledgements

The authors appreciate financial supports of NSF Award No. IIS-0117733, NIST MEL Award No. 60NANB-2D0137, PTI/NIST Award No. AGR20020506, and NIST Award No. SB 1341-03-W-0338.

References

[1] Meystel, A. M., and Albus, J. S. "Intelligent Systems: Architecture, Design and Control", *John Wiley & Sons, Inc., New York*, 2002.

[2] Mayne, D. Q., Rawlings, J. B., Rao, C.V., and Scokaert, P.O.M. "Constrained Model Predictive Control: Stability and Optimality", *Automatica*, Vol. 36, 2000.

[3] Mayne, D. Q. "Control of Constrained Dynamic Systems", *European Journal of Control*,7, 2001.

[4] Henson, M. A. "Nonlinear Model Predictive Control: Current Status and Future Directions", *Computers and Chemical Engineering*,23, 1998.

[5] Mahadevan, R., Doyle III, F.J. "Efficient Optimization Approaches to Nonlinear Model Predictive Control", *International Journal of Robust and Nonlinear Control (submitted)*, 2002.

[6] Longo, E., Teppati, G., and Bellomo, N. "Discretization of Nonlinear Models by Sinc Collocation-Interpolation Methods.", *Computers Math. Applic.*, Vol. 32, No. 4, 1996.

[7] Proakis, J. G., and Manolakis, D. G. "Digital Signal Processing", *Prentice Hall, New Jersey.*, 1996.

[8] Meijering, E. H. W., Niessen, W., J., and Viergever, M. A. "The Sinc-Approximating Kernel of Classical Polynomial Interpolation.", *IEEE Int. Conf. on Image Processing, Los Alamitos, CA*, Vol. 3, 1999.

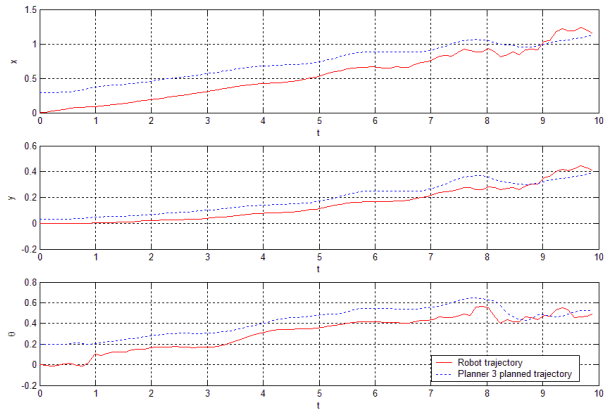

Figure 6: Experiment: The robot starts at an offset from the initial plan but catches up and reaches the goal.

[9] Faiz, T. N., Agrawal, S. K. and Murray, R. "Trajectory Planning of Differentially Flat Systems with Dynamics and Inequalities", *J. Guid. Control Dynamics*, 24(2), 2000.

[10] Murray, R.M., and Nieuwstadt, M.J.V. "Real-Time Trajectory Generation for Differentially Flat Systems", *International Journal of Robust and Nonlinear Control*,8, 1998.

[11] Biegler, L. T. "Solution of Dynamic Optimization Problems by Successive Quadratic Programming and Orthogonal Collocation.", *Computers and Chemical Engineering*, Vol. 8, No. 3, 1984.

[12] Press William H. "Numerical Recipes in C : The Art of Scientific Computing", *Cambridge University Press; 2nd edition*, January 1993

[13] NIST Manufacturing Engineering Lab. "Real-Time Control Systems Library", *http://www.isd.mel.nist.gov/projects/rcslib/*

[14] Sandia National Laboratories. "OPT++: An Object-Oriented Nonlinear Optimization Library", *http://csmr.ca.sandia.gov/projects/opt/opt++.html*